IN THE SPECIFICATION

Please amend the paragraph beginning on page 3, line 29 as follows:

Example embodiments of the invention are directed to a system and method for distributing software. The detailed description is divided into [[five]] six sections. In the first section, terms are defined. In the second section, the hardware and the operating environment, in conjunction with which embodiments of the invention can be practiced, are described. In the third section, a system level overview of some embodiments of the invention is presented. In the fourth section, methods of using example embodiments of the invention are provided. In the fifth section, particular implementations of the invention are described according to one embodiment. In the final section, several alternate embodiments and examples are described.

Please amend the paragraph beginning on page 5, line 28 as follows:

Computer 100 comprises a central processing unit 108, random-access memory (RAM) 110, read-only memory (ROM) 112 and one or more storage devices 114. The central processing unit 108 represents a central processing unit of any type of architecture, such as CISC (Complex Instruction Set Computer), RISC (Reduced Instruction Set Computer) or the like. Although computer 100 is shown having only a single processor, embodiments of the present invention also apply to computers having multiple processors. RAM 110 and ROM 112 are collectively referred to as the memory of computer 100. The one or more storage devices 114 represent mechanisms for storing data. The one or more storage devices 114 may include magnetic disk storage media, optical storage media, and/or other machine-readable media. Although only one storage device 114 is shown, multiple storage devices and multiple types of storage devices may be present. In some ~~embodiment~~ embodiments, the storage devices 114 may be distributed across other electronic devices attached to a network. In one embodiment, storage device 114 stores one or more channels for a software distribution system as further described below.

Please amend the paragraph beginning on page 7, line 14, by inserting a comma (,) in line 15 as follows:

In one embodiment, each task is also associated with instructions for "rolling it back," in other words, doing the reverse of executing it. In the case of deploying a file, this is the act of "uninstalling" that same file. In the case of the other types of tasks, the administrator can optionally define a script or custom commands that are to be used to "roll back" the task.

Please amend the paragraph beginning on page 9, line 3 as follows (including inserting a comma (,) in line 13):

The "task" and "target" objects are divided into two separate root nodes to avoid ambiguity between existence and assignment. For instance, if a task "spreadsheet" were located in the "marketing" container, a tree view that was not divided into separate root nodes would make it impossible to determine whether the task "spreadsheet" *belonged* to the container "marketing" or was *assigned* to it. The latter would mean that the task was merely placed in the container for convenience of organization, while the former would mean that every computer or user in the container would receive the task. Another motivation for the parallel "tasks" and "targets" nodes is to make administration using drag-and-drop more effective. An alternate~~,~~ ~~rejected~~ design would be to force all tasks to be created at the "root" level; in other words they would all be parallel to one another, and they could not belong to a container.

Please amend the paragraph beginning on page 11, line 3 as follows:

In one embodiment, there can be any number of channel consoles 202 using the same channel at the same time. It is the channel server's 204 responsibility to coordinate the changes they make with each other. The channel server 204 maintains the one or more channels and the deployment instructions

Please amend the paragraph beginning on page 11, line 7 as follows:

In one embodiment, the channel server 204 is a Windows executable program, service, and/or module without a visible interface. The channel server 204 communicates information about the database to the channel console 202 and channel client 206 components, and makes changes to the database dictated by the channel console 202. In one embodiment, there is only one channel server 204 for each channel database 210, but the reverse is not necessarily true; one

channel server 204 can control multiple channel databases. The channel server 204 is not a visible component; it is installed and manipulated solely via the channel console 202.

Please amend the paragraph beginning on page 12, line 5 as follows:

The channel console 202, the channel server 204 and the channel client [[208]] 206 are computers such as the computer 100 shown in Figure 1. Although not depicted, in some embodiments the channel console 202 and the channel server 204 are a single computer.

Please amend the paragraph beginning on page 15, line 15 as follows:

Figure 3 is a block diagram of the processing modules for an example embodiment of the system of Figure 2. The computer system comprises a channel console component 302 to provide a user interface to create one or more channels 306 comprising a list of targets and tasks, and a channel server component 304 to store the one or more channels, wherein at least one of the tasks is stored in a different location than the channel. The channel server component 304 also provides deployment [[309]] instructions 309 for the tasks in the one or more channels 306. In one embodiment, the list of targets comprises computers, users, groups or containers. In one embodiment, the list of tasks comprises files (or packages), scripts and commands.

Please amend the paragraph beginning on page 16, line 3 as follows:

Figure 5 is a block diagram of a software distribution system according to an alternate embodiment of the invention. The software distribution system shown in Figure 5 comprises a channel server 204 and a channel client 206. In one embodiment, the channel server 204 [[to]] stores the one or more channels 306 in a channel database 210. The channel database 210 also stores deployment instructions 309 for a channel. In one embodiment, the deployment instructions 309 include a schedule and a task location. In the example embodiment [[show]] shown in Figure 5, the deployment file location 310 is a different location than the channel server 204.

Please amend the paragraph beginning on page 24, line 28 as follows:

Figure 10C is a flow chart of an example embodiment of a method of executing a task. As shown in Figure 10C, it is first determined if a special Windows account is required to execute the task (block 1028). If a special Windows account is required, then the special account is used to log on to the channel client 206 (block 1030). If the logon is unsuccessful (block 1032), then a deployment report is created (block [[1035]] 1034). If the logon is successful (block 1032), then it is determined if the task to be executed is a package (block 1036).

Please amend the paragraph beginning on page 25, line 4 as follows:

If the task is a package, then the package is installed (block 1038). If the installation of the package is unsuccessful (block 1040), then a deployment report is created indicating that the installation of the package on the channel client 206 failed (block 1042). Otherwise, if the installation of the package is successful (block 1040), then a deployment report is created indicating that the installation of the package on the channel client 206 succeeded (block 1041).

Please amend the paragraph beginning on page 25, line 10 as follows:

If the task is a script (block 1044), then the script is played (block 1046). If the installation of the script is unsuccessful (block 1048), then a deployment report is created indicating that the installation of the script on the channel client 206 failed (block 1050). Otherwise, if the installation of the script is successful (block 1048), then a deployment report is created indicating that the installation of the script on the channel client 206 succeeded (block 1052).

Please amend the paragraph beginning on page 25, line 16 as follows:

In one embodiment, if the task is not a package or a script, then the task is a command to execute (block 1054). If the execution of the command is unsuccessful (block 1056), then a deployment report is created indicating that the execution of the command on the channel client 206 failed (block 1058). Otherwise, if the execution of the command is successful (block [[1066]] 1056), then a deployment report is created indicating that the execution of the command on the channel client 206 succeeded (block 1060).

Please amend the paragraph beginning on page 27, line 1 as follows (including deletion of a period on page 28, line 15):

Figure 12 is a more detailed flow chart of an example embodiment of a method of directly installing the channel client component software on a computer that is directly connected to a network. In one embodiment, the method shown in Figure 12 is initiated by a channel console to install the channel client component software on a target computer. However, in an alternate embodiment, the channel server initiates the method. In one embodiment, the channel console creates a list of computers to which files will be directly installed (block 1202) and selects whether the computers will be rebooted automatically (block 1204). For each computer to be directly installed (block 1206), it is determined whether the target computer is the same one running the algorithm (i.e. installing "locally" vs. "remotely") (block 1218). If the installation is remote (block 1218), the channel console attempts to establish a remote registry connection to the target computer (block 1220). If the registry connection is not established (block 1222), then determine if a fatal error occurred (block 1216). If a fatal error has occurred ~~(block 1214)~~ (block 1216), cancel the remaining installation instructions (block 1214). If a fatal error has not occurred (block 1216), then determine if the user wishes to reattempt the installation instructions (block 1212). If the user wishes to reattempt the installation, the user fixes the problems that prevented the installation (block 1210). If a remote registry connection is established (~~block1222~~ block 1222), then it is determined whether the target computer uses service applications (block 1224). For a remote installation (block 1226), copy the client and the service files to be installed to the default file share, C$, of the target computer (block 1228). If it is not a remote installation (block 1226), create the channel registry entries on the target computer. Remove any existing channel service registry entries from the target computer (block 1230). Then, for a remote installation (block 1232), using remote service management, attempt to register the channel service located at C$ of the target computer. If successful, start the channel service (block 1234). If the channel service registration is successful (block 1236), create temporary registry entries for the channel service that describe permanent locations of the client files (block 1238). The channel service launches the client on the target computer with the command to install itself (block 1240). The client application copies client and service files to the appropriate folders, creates or updates registry entries, launches the service applications, if appropriate, then deletes

the source files from C$ and reboots the target computer if instructed (block 1242). If the target computer was not instructed to reboot (block 1244), the client establishes a connection to the channel server upon next reboot of the target computer (block 1246). If the target computer was instructed to reboot (block 1244), then it is determined if more computers are to be installed (block 1248). If no more computers are to be installed (block 1248), then the direct installation is complete (block 1249). Returning back to block 1236, if the channel service registration was not successful, then a temporary registry entry is created that will invoke client installation upon next reboot of the target computer (block 1250). Returning now to block 1232, if the installation is local, then the client on the local computer is launched with the command to install itself (block 1252). The client application then copies client and service files to the appropriate folders, creates or updates registry entries, launches the service applications, if appropriate, and connects to the channel server[[.]] (block 1254). If no more computers are to be installed (block 1248), then the direct installation is complete (block 1249).

Please amend the paragraph beginning on page 28, line 17 as follows:

Figure 13 is a more detailed flow chart of an example embodiment of a method of installing the channel client component software on a target computer that is not directly connected to a network. In the example embodiment shown in Figure 13, subscription files are used. In one embodiment, the subscription files are authored by a channel console and read by channel client. However, in alternate embodiments, the channel server authors subscription files. As shown in Figure 13, the subscription file is opened and read by a channel client (block 1302). It is determined whether or not a prompt needs to be displayed (block 1304) and, if yes, then the user is prompted regarding whether or not to install the subscription file (block 1306). If the user accepts the installation of the subscription file (block 1308), then it is determined whether or not the client needs to be installed or uninstalled (block 1310). If the client needs to be installed (block 1312), it is verified that the source file for the installation exists (block 1314). To install the client, the appropriate destination folders are determined and created if necessary, the client and service files are copied, registry entries are updated or created, and the service application is launched (block 1314). If the client is to be uninstalled (block 1312), the location of the existing client and service files are determined, the service applications are stopped, the client and service

files are deleted, the registry entries are removed or revised, and client folders are deleted if appropriate (block 1316). The channel client then determines if the channel needs to be installed or uninstalled (block 1318). If the channel needs to be installed (block 1320), the channel registry entries are created or updated, and a connection to the channel server is opened (block 1322). If the channel needs to be uninstalled (block 1320), the connection to the channel server is closed and, if necessary, the channel registry entries are removed (block 1324). Then, the channel client determines if the subscription file needs to be deleted (block 1326). If yes, then the subscription file is deleted (block 1328) and the method is completed (block 1330).

Please amend the paragraph beginning on page 39, line 11 as follows:

Automatic Installation and Updates. As described above, when a channel is created, a link is automatically made available for subscribing to the channel and installing the client if it is not already installed. When the console is installed on a computer, an installation shell extension is installed with it. This shell extension allows the administrator to install the client by making a selection from the context menu associated with a computer in the operating system's "Network Neighborhood" area (this area has different names in the different ~~flavors~~ versions of Windows) or in the network management area of Windows 2000.

Please amend the paragraph beginning on page 39, line 24 as follows:

The channel server 204 stores in the channel database 210 all the files needed for the channel client 206 and channel console 202 to execute. Every time the client contacts the channel server 204, a check is made to ensure the client has the latest files. If it does not, the updated files are copied to the client, which then resumes execution without the need for a reboot. Similarly, every time a console contacts the channel server 204, it ensures the server has the latest versions of the same files, that the server itself is up to date, and that the console is up to date. When Prism Deploy is installed the latest files for the client, server, and console are automatically installed locally along with everything else. Thus once the original media or files directly from Lanovation are installed, all other components are automatically updated using the channel itself.